案例 18-直线中点分割直线段裁剪算法

文档编写：霍波魏

校稿/修订：孔令德

时间 2019~2020

联系方式：QQ997796978

说明：本套案例由孔令德开发，原版本为 Visual C++6.0，配套于孔令德的著作《计算机图形学-基于 MFC 三维图形开发》一书。孔令德计算机工程研究所的学生霍波魏在学习计算机图形学期间，对本套案例进行了升级并编写了学习文档。现在程序的编写和程序的解释都是基于 Windows 10 操作系统，使用 Microsoft visual studio 2017 平台的 MFC（英文版）开发。

一、知识点

中点计算公式：对于端点坐标为$P_0(X_0,Y_0)$和$P_1(X_1,Y_1)$的直线段，中点坐标的计算公式为：

$$P = (P_0, P_1)/2$$

展开形式为

$$X = (X_0, X_1)/2$$

$$Y = (Y_0, Y_1)/2$$

二、案例描述

本案例使用直线中点分割直线段裁剪算法，通过绘制矩形框对金刚石进行裁剪。

三、实现步骤

1. 添加绘制直线类 Cline 类。
2. 在 CTestView 中添加编码函数、绘制裁剪窗口函数、中点分割函数、坐标变换、裁剪函数以及金刚石绘制函数。
3. 在 CTestView 中添加消息函数，在 OnDraw 中调用 DoubleBuffer 函数定义客户区坐标、绘制裁剪窗口。

四、主要算法

CTestView 类

```
public:
```

```
    void DoubleBuffer(CDC* pDC);//双缓冲
    void DrawWindowRect(CDC* pDC);//绘制裁剪窗口
    BOOL CSLineClip();//裁剪算法
    void EnCode(CP2d &pt);//编码函数
    CP2i Convert(CPoint point);//坐标系转换
    void Diamond(CDC* pDC);//绘制金刚石函数
    void MidClip(CP2d p0, CP2d p1);//中点分割函数
protected:
    int nClientWidth, nClientHeight;//屏幕客户区宽度和高度
    int nHWidth, nHHeight;//屏幕客户区的半宽和半高
    CLine* line;//直线的指针
    CP2d P[2];//直线的起点和终点
    int RtCount;//窗口顶点个数
    BOOL bDrawRect;//是否允许画线
    BOOL bClip;//是否裁剪
    CP2d *V;//动态定义等分点数组
    CP2i Rect[2];//定义裁剪窗口对角坐标
#define LEFT      0x0001   //代表0001
#define RIGHT     0x0002   //代表0010
#define BOTTOM    0x0004   //代表0100
#define TOP       0x0008   //代表1000
void CTestView::DoubleBuffer(CDC* pDC)//双缓冲
{
    CRect rect;//定义客户区
    GetClientRect(&rect);//获得客户区的大小
    nClientWidth = rect.Width();//屏幕客户区宽度
    nClientHeight = rect.Height();//屏幕客户区高度
    nHWidth = nClientWidth / 2;//屏幕客户区半宽
    nHHeight = nClientHeight / 2;//屏幕客户区半高
    CDC memDC;
    memDC.CreateCompatibleDC(pDC);
    CBitmap NewBitmap, *pOldBitmap;
    NewBitmap.CreateCompatibleBitmap(pDC, nClientWidth, nClientHeight);
    pOldBitmap = memDC.SelectObject(&NewBitmap);
    memDC.FillSolidRect(&rect, pDC->GetBkColor());
    if (RtCount && !bClip)
        DrawWindowRect(&memDC);//绘制窗口
    Diamond(&memDC);//绘制金刚石
    pDC->BitBlt(0, 0, nClientWidth, nClientHeight, &memDC, 0, 0,
SRCCOPY);//将内存位图拷贝到屏幕
    memDC.SelectObject(pOldBitmap);//恢复位图
    NewBitmap.DeleteObject();//删除位图
}
void CTestView::DrawWindowRect(CDC* pDC)//绘制裁剪窗口
```

```
{
    CRGB LineClr = CRGB(0.0, 0.5, 0.0);
    line->MoveTo(pDC, nHWidth + Rect[0].x, nHHeight - Rect[0].y, LineClr);
    line->LineTo(pDC, nHWidth + Rect[1].x, nHHeight - Rect[0].y, LineClr,
3);
    line->LineTo(pDC, nHWidth + Rect[1].x, nHHeight - Rect[1].y, LineClr,
3);
    line->LineTo(pDC, nHWidth + Rect[0].x, nHHeight - Rect[1].y, LineClr,
3);
    line->LineTo(pDC, nHWidth + Rect[0].x, nHHeight - Rect[0].y, LineClr,
3);
}
BOOL CTestView::CSLineClip()//裁剪算法
{
    EnCode(P[0]);//起点编码
    EnCode(P[1]);//终点编码
    while (P[0].rc != 0 || P[1].rc != 0)//至少有一个顶点在窗口外
    {
        if ((P[0].rc&P[1].rc) != 0)//简弃之
        {
            RtCount = 0;
            return FALSE;
        }
        if (0 == P[0].rc)//P[0]在窗口之外
        {
            CP2d pTemp;
            pTemp = P[0];
            P[0] = P[1];
            P[1] = pTemp;
        }
        MidClip(P[0], P[1]);
    }
}
void CTestView::EnCode(CP2d &pt)//编码函数
{
    pt.rc = 0;
    if (pt.x < Rect[0].x)
        pt.rc = pt.rc | LEFT;
    else if (pt.x > Rect[1].x)
        pt.rc = pt.rc | RIGHT;
    if (pt.y < Rect[1].y)
        pt.rc = pt.rc | BOTTOM;
    else if (pt.y > Rect[0].y)
        pt.rc = pt.rc | TOP;
```

```
}
CP2i CTestView::Convert(CPoint point)//坐标系转换
{
    CP2i ptemp;
    ptemp.x = point.x - nHWidth;
    ptemp.y = nHHeight - point.y;
    return ptemp;
}
void CTestView::Diamond(CDC* pDC)//绘制金刚石函数
{
    double thta;//thta为圆的等分角
    int n = 20;//定义等分点个数
    V = new CP2d[n];
    double r = 300;//定义圆的半径
    thta = 2 * PI / n;
    for (int i = 0; i < n; i++)//计算等分点坐标
    {
        V[i].x = r * cos(i * thta);
        V[i].y = r * sin(i * thta);
    }
    for (int i = 0; i <= n - 2; i++)//依次连接各等分点
    {
        for (int j = i + 1; j <= n - 1; j++)
        {
            if (!bClip)
            {
                line->MoveTo(pDC, ROUND(nHWidth + V[i].x),
                    ROUND(nHHeight - V[i].y));
                line->LineTo(pDC, ROUND(nHWidth + V[j].x),
                    ROUND(nHHeight - V[j].y));
            }
            else
            {
                P[0] = V[i];//对金刚石的每段直线进行裁剪
                P[1] = V[j];
                if (CSLineClip())
                {
                    line->MoveTo(pDC, nHWidth + ROUND(P[0].x),
                                nHHeight - ROUND(P[0].y));
                    line->LineTo(pDC, nHWidth + ROUND(P[1].x),
                                nHHeight - ROUND(P[1].y));
                }
            }
        }
```

```
    }
    delete[]V;
}
void CTestView::MidClip(CP2d p0, CP2d p1)//中点分割函数
{
    CP2d p;//中点坐标
    p = (p0 + p1) / 2;
    EnCode(p);
    while (fabs(p.x - p0.x) > 1e-6 || fabs(p.y - p0.y) > 1e-6)//判断结束
    {
        if (0 == p.rc)//中点也在窗口内，舍弃P0
            p1 = p;
        else//舍弃P1
            p0 = p;
        p = (p0 + p1) / 2;
        EnCode(p);
    }
    P[0] = p;
}
```

五、实现效果

直线中点分割直线段裁剪算法效果如图 18-1、18-2 和 18-3 所示。

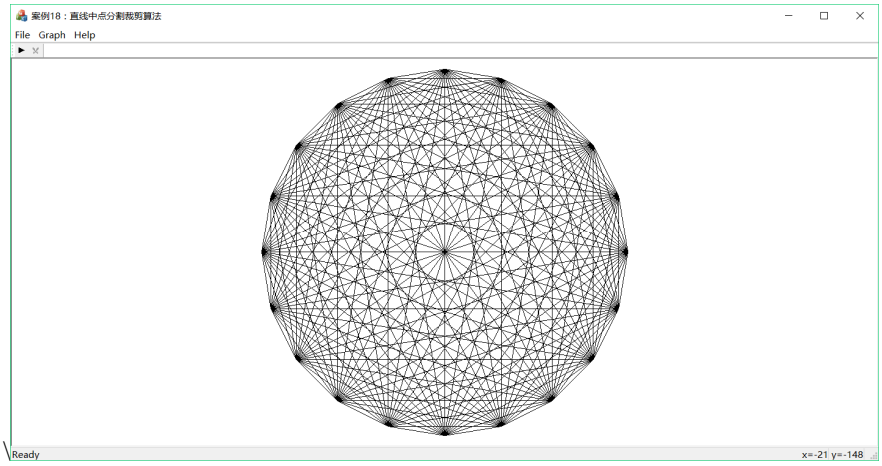

图 18-1 直线中点分割直线段裁剪算法效果图一

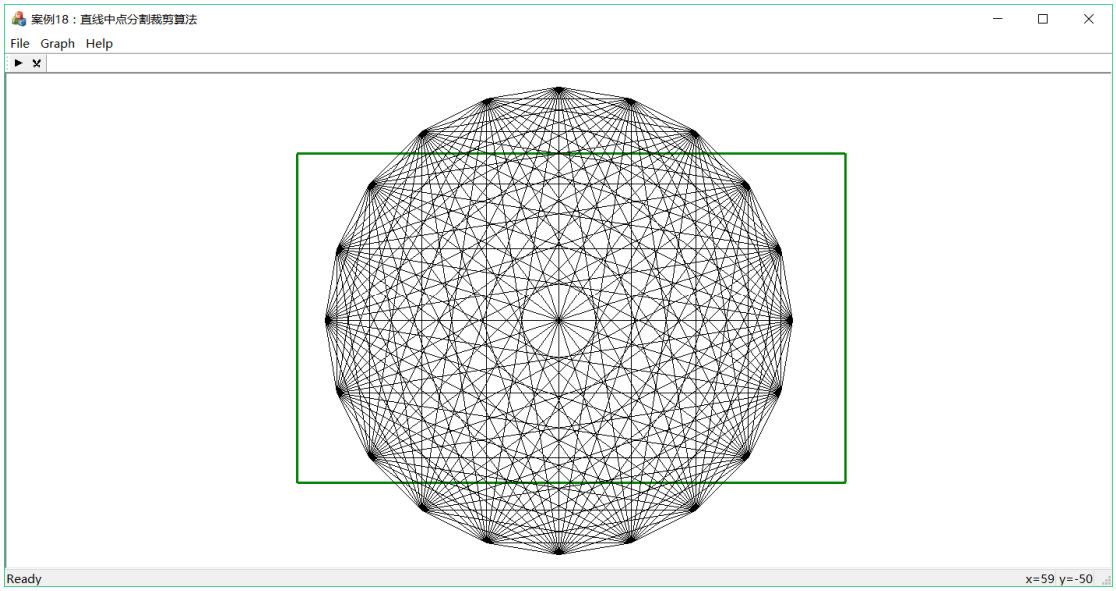

图 18-2 直线中点分割直线段裁剪算法效果图二

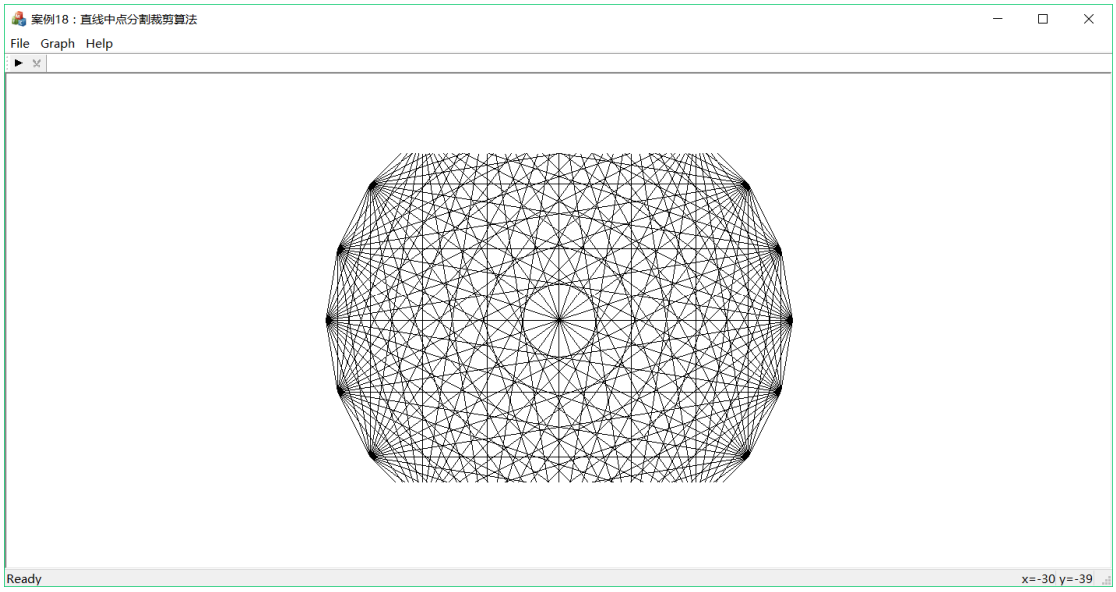

图 18-3 直线中点分割直线段裁剪算法效果图三

六、补充

Cohen-Sutherland 算法和中点分割直线段裁剪算法的区别：两者都是对直线段端点进行编码，并把直线段与位置关系分为三种情况。前两种情况都采用“简取”和“简弃”进行简单处理，而第三种情况不同，Cohen-Sutherland 算法是通过只需按段与窗口边界求交点，而中点分割直线段裁剪算法是利用二分算法的思想，将直线段分成两段，对每一段进行“简取”和“简弃”处理，对于不能处理的直线段再继续分下去，直到每段直线完全位于窗口之内或者窗口之外。